UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/784,125	02/20/2004	Douglas J. Koslow	CA7037682001	5265

55497 7590 12/19/2008
VISTA IP LAW GROUP LLP
1885 Lundy Avenue
Suite 108
SAN JOSE, CA 95131

EXAMINER	
JACOB, MARY C	

ART UNIT	PAPER NUMBER
2123	

MAIL DATE	DELIVERY MODE
12/19/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/784,125	KOSLOW ET AL.	
	Examiner	Art Unit	
	MARY C. JACOB	2123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *23 October 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-38* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-38* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. The response filed 10/23/08 has been received and considered. Claims 1-38 have been presented for examination.

Continued Examination Under 37 CFR 1.114

2. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 10/23/08 has been entered.

Claim Objections

3. Claims 27-30 are objected to because of the following informalities. Appropriate correction is required.

4. Claim 27 is directed to “A computer program product comprising a volatile or non-volatile computer usable medium having executable code to execute a process for debugging of an electronic design comprising both an HDL portion and a general programming language portion, the process comprising…”. This preamble would be better if written such that it reflects that the medium is executed by a computer, for example, "...having computer executable code, that when executed by a computer,

causes the computer to execute a process for debugging…", as discussed in the specification in reference to Figure 5.

5. Claim 27, line 7 recites, "debut", it should read, "debug".

6. Claim 28, line 11 recites, "the request for processing function", it would be better if written, "the request processing function".

7. Claim 29 is directed to "A computer program product comprising a volatile or non-volatile computer usable medium having executable code to execute a method...the method comprising...". This preamble would be better if written such that it reflects that the medium is executed by a computer, for example, "…computer usable medium having executable code, that when executed by a computer, causes the computer to execute a method…" as discussed in the specification in reference to Figure 5.

8. Claim 30, lines 19-20 recite, "the request for processing function", it would be better if written, "the request processing function".

Claim Rejections - 35 USC § 101

9. The rejections of Claims 28, 30, 33, 34, 37 and 38 under 35 U.S.C. 101 recited in the 6/23/08 Office Action, have been withdrawn in view of the amendments to the claims, filed 10/23/08.

10. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

11. Claims 1-26 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Claims 1 and 18 are directed to methods, however, these methods are not patent eligible processes under 35 U.S.C. 101 because they are either not tied to another statutory class (such as a particular apparatus) or do not transform underlying subject matter (such as an article or materials) to a different state or thing. Although the final steps in these claims set forth that the *results* of the method are stored in a computer-readable medium, the method itself is not tied to another statutory class or does not transform underlying subject matter to a different state or thing. For example, the claim does not identify the apparatus that accomplishes the method steps.

Claim Rejections - 35 USC § 102

12. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

13. **Claims 1-7, 12, 13, 15-17, 27, 28, 31-34** are rejected under 35 U.S.C. 102(b) as being anticipated by Hollander (US Patent 6,182,258).

14. As to **Claims 1, 27 and 28**, Hollander teaches: a method (column 6, lines 12-13), system comprising means (column 6, lines 12-13; Figure 1; Figure 3) and computer program product comprising a volatile or non-volatile computer usable medium having executable code (Figure 1, "Runtime Environment", Figure 3, element 58, "basic

executable"; column 9, lines 26-40, specifically, wherein the executable is "run", therefore the executable must be stored on a computer usable medium in order to be run in the runtime environment) for debugging an electronic design (column 10, lines 39-43, lines 50-58; column 11, lines 2-5) having both an HDL portion (Figure 5, element 170; column 10, lines 35-36) and a general programming language portion (Figure 5, elements 172, 163; column 10, lines 24-28; column 11, lines 30-32), comprising: interrupting a simulator that operates upon the HDL portion of the electronic design to allow for debugging of the HDL portion (column 5, lines 44-48; column 9, lines 12-13; column 10, lines 35-36 and lines 44-56), the simulator interrupted by an external debugger (column 5, lines 44-48; column 10, lines 44-50; Figure 6, element 90, "Stop on errors", "Breakpoints") to debug the general programming language portion of the electronic design (column 10, lines 24-28, lines 43-46 and 50-61; column 11, lines 2-16); handling a simulator request with the external debugger for the simulator that is interrupted, the external debugger calling a request processing function at the simulator, the simulator request for simulation of the HDL portion (column 5, lines 44-48; column 9, lines 58-65; column 10, lines 43-50; column 11, lines 6-13; Figure 6, element 90, "Continue", "Return"); executing the request processing function at the simulator to respond to the simulator request , wherein the means for executing the request processing function comprises a processor (column 9, lines 58-65; column 10, lines 47-49; column 9, lines 26-40, specifically, the executable of the invention is "run", wherein it is understood that the executable is run by a processor); and generating debug results based upon executing the request processing function and storing the debug results in

a computer-readable medium (column 9, lines 21-24; column 10, line 65-column 11, line 11, wherein it is understood that the debug results are stored in a computer-readable medium in order for the results to be displayed and to allow the simulation to be recorded and replayed; Figure 1, element 24).

15. As to Claims 2, 31 and 33, Hollander teaches: the simulator request accesses a portion of the HDL portion (column 7, lines 27-30; column 13, lines 38-41).

16. As to Claim 3, Hollander teaches: the simulator request accesses HDL signal values (column 9, lines 58-65).

17. As to Claim 4, Hollander teaches: the simulator request accesses HDL design hierarchy (column 7, lines 27-30; column 9, lines 59-65).

18. As to Claim 5, Hollander teaches: the simulator request operates simulator functionality (column 9, lines 59-65; column 10, lines 46-49; Figure 6, element 90, “continue”, “stop on errors”, “breakpoints”).

19. As to Claims 6, 32 and 34, Hollander teaches: the general programming language portion comprises C, C++, or SystemC code (Hollander: column 10, lines 24-28; column 11, lines 30-32).

20. As to Claim 7, Hollander teaches: the HDL portion comprises VHDL or Verilog (Hollander: column 6, lines 37-39 and lines 61-63; column 10, lines 34-35).

21. As to Claim 12, Hollander teaches: the simulator request is generated at a simulator GUI (Hollander: column 10, line 59-column 11, line 8, Figure 6, element 90).

22. As to Claim 13, Hollander teaches: the response to the simulator request is displayed at the simulator GUI (Hollander: column 10, line 59-column 11, line 8, Figure 6).

23. As to Claims 15, Hollander teaches: the simulator request is routed through a debugger GUI for the external debugger (Hollander: column 10, line 59-column 11, line 8, Figure 6).

24. As to Claim 16, Hollander teaches: the simulator request is directly routed to the external debugger (Hollander: column 10, line 59-column 11, line 8, Figure 6).

25. As to Claim 17, Hollander teaches: the request processing function is set up ahead of time at the simulator to handle anticipated simulator requests (Hollander: column 5, lines 44-48; column 11, lines 6-8).

Claim Rejections - 35 USC § 103

26. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each

claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

27. **Claims 8-11** are rejected under 35 U.S.C. 103(a) as being unpatentable over Hollander as applied to claim 1 above, in view of Chan (US Patent 6,466, 898).

28. Hollander teaches a method for simultaneous debugging of an electrical design wherein a simulator request is handled with an external debugger, the external debugger calling a request processing function at a simulator, the simulator request for simulation of the HDL portion of the design.

29. Hollander does not expressly teach (claim 8) the action of having the external debugger call the request processing function is based upon recognition of a waiting simulator request, (claim 9) recognition of the waiting simulator request is based upon a message sent to the external debugger, (claim 10) recognition of the waiting simulator request is based upon a periodic check of a simulator request wait queue, (claim 11) recognition of the waiting simulator request is based on whether a threshold number of simulator requests are waiting in a simulator request wait queue.

30. Chan teaches a novel concurrent, multi-threaded algorithm to accelerate the execution of logic simulation of HGL designs that supports both VHDL and Verilog HDL design languages in a single platform (column 4, lines 5-16), wherein the execution of logic simulation events includes wherein (claim 8) the call of a request processing function is based upon recognition of a waiting simulator request (column 7, lines 1-7, lines 32-39; Figure 3, element 14), (claim 9) wherein the recognition of the waiting

simulator request is based upon a message sent to the logic simulation program (column 7, lines 1-7, lines 32-39; Figure 3, element 14), wherein (claim 10) recognition of the waiting simulator request is based upon a periodic check of a simulator request wait queue (column 7, lines 1-7, lines 32-46; Figure 3, element 14), and wherein (claim 11) recognition of the waiting simulator request is based on whether a threshold number of simulator requests are waiting in a simulator request wait queue (column 20, lines 55-66; Figure 17, element 89).

31. Hollander and Chan are analogous art since they are both directed to the simulation of a HDL design.

32. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the method for simultaneous debugging of an electrical design wherein an external debugger calls a request processing function at a simulator as taught by Hollander to further include wherein the call of a request processing function is based upon recognition of a waiting simulator request, wherein the recognition of the waiting simulator request is based upon a message sent to the logic simulation program, wherein recognition of the waiting simulator request is based upon a periodic check of a simulator request wait queue, and wherein recognition of the waiting simulator request is based on whether a threshold number of simulator requests are waiting in a simulator request wait queue as taught in Chan since Chan teaches a novel concurrent, multi-threaded algorithm to accelerate the execution of logic simulation of HGL designs that supports both VHDL and Verilog HDL design languages in a single platform (column 4, lines 5-16).

33. **Claim 14** is rejected under 35 U.S.C. 103(a) as being unpatentable over Hollander as applied to claim 1 above, in view of Stallman et al ("Debugging with GDB: The GNU Source-Level Debugger", January 2002, book summary, obtained on www.gnu.org).

34. As to Claim 14, Hollander teaches an external debugger debugging the general programming language portion of a design, the external debugger calling a request processing function at the simulator.

35. Hollander does not expressly teach: the external debugger is a gdb debugger.

36. Stallman et al teaches the gdb, the GNU source level debugger that supports C and C++ among other languages, allows a designer to see what is going on inside a program while it executes or what a program was doing the moment it crashed and allows a designer to catch bugs in a program by: starting the program and specifying anything that might effect the program's behavior, making the program stop under specified conditions, examining what happened when the program stopped and allowing the designer to experiment with changes to see what effect they have on the program (paragraphs 1-3, bullets 1-4).

37. Hollander and Stallman et al are analogous art since they are both directed to the debugging of a general programming language portion of a design.

38. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the external debugger debugging the general programming language portion of the design as taught in Hollander to include the

external debugger being a gdb debugger since Stallman et al teaches that a gdb debugger allows a designer to see what is going on inside a program while it executes or what a program was doing the moment it crashed and allows a designer to catch bugs in a program by: starting the program and specifying anything that might effect the program's behavior, making the program stop under specified conditions, examining what happened when the program stopped and allowing the designer to experiment with changes to see what effect they have on the program (paragraphs 1-3, bullets 1-4).

39. **Claims 18-23, 25, 26, 29, 30, 36, and 38** are rejected under 35 U.S.C. 103(a) as being unpatentable over Hollander in view of Chan.

40. As to **Claims 18, 29 and 30**, Hollander teaches: a method (column 6, lines 12-13), system comprising means (column 6, lines 12-13; Figure 1; Figure 3) and computer program product comprising a volatile or non-volatile computer usable medium having executable code (Figure 1, "Runtime Environment", Figure 3, element 58, "basic executable"; column 9, lines 26-40, specifically, wherein the executable is "run", therefore the executable must be stored on a computer usable medium in order to be run in the runtime environment) for processing of a design (column 10, lines 39-43, lines 50-58; column 11, lines 2-5) that is based upon multiple programming languages, the design comprising a first language portion (Figure 5, element 170; column 10, lines 35-36) and a second language portion (Figure 5, elements 172, 163; column 10, lines 24-28; column 11, lines 30-32), the method comprising: processing the second language portion of the design to cause an interruption of processing for the first language

portion, wherein the processing for the first language portion is interrupted to process the second language portion (column 5, lines 44-48 “breakpoints”; column 9, lines 58-65; column 10, lines 12-13, lines 44-50, “The external program runs at full speed until it reaches pre-designated points at which the program interacts with the DUT… the invention interprets the co-verification request and executes the appropriate functions", “The invention 162 can drive 32 or sample 24 and simulator signals…Verilog tasks or VHDL procedures can also be called”); indicating a need for processing of the second language portion to call a request processing function at the first language portion (column 9, lines 58-65; column 10, lines 44-50, “The external program runs at full speed until it reaches pre-designated points at which the program interacts with the DUT. A co-verification request is sent through the socket to the invention"); having the processing of the second language portion call a request processing function at the first language portion that has been interrupted (column 5, lines 44-48; column 9, lines 58-62; column 10, lines 43-50; Figure 6, element 90, “Continue”); the request for processing of the first language portion causes the processing of the first language portion (column 9, lines 58-65, “The invention 162 can drive…any simulator 36 signals. Verilog tasks or VHDL procedures can also be called”); executing the request processing function at the first language portion to process the request (column 9, lines 58-65; column 10, lines 47-49) wherein the means for executing the request processing function comprises a processor (column 9, lines 58-65; column 10, lines 47-49; column 9, lines 26-40, specifically, the executable of the invention is “run”, wherein it is understood that the executable is run by a processor); and generating processing results based upon executing the request

processing function and storing the processing results in a computer-readable medium (column 9, lines 21-24; column 10, line 65-column 11, line 11 wherein it is understood that the debug results are stored in a computer-readable medium in order for the results to be displayed and to allow the simulation to be recorded and replayed; Figure 1, element 24).

41. Hollander does not expressly teach: determining whether there are one or more waiting requests for processing of the first language portion.

42. Chan teaches a novel concurrent, multi-threaded algorithm to accelerate the execution of logic simulation of HGL designs that supports both VHDL and Verilog HDL design languages in a single platform (column 4, lines 5-16), wherein event-driven logic simulation, as known in the art and by the invention as taught in Chan, includes determining whether there are one or more waiting requests for processing of a portion of the mixed language design (column 7, lines 1-7, lines 32-39; Figure 3, element 14; Figure 8, element 42).

43. Hollander and Chan are analogous art since they are both directed to the simulation of a mixed language design.

44. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the simultaneous processing of a design based on multiple programming languages as taught by Hollander to further include determining whether there are one or more requests waiting for processing a portion of the design as taught in Chan since Chan teaches a novel concurrent, multi-threaded algorithm to accelerate the execution of logic simulation of HGL designs that supports both VHDL

and Verilog HDL design languages in a single platform (column 4, lines 5-16) and further teaches that event driven simulation, as known in the art, includes determining whether there are one or more waiting requests for processing of a portion of the mixed language design (column 7, lines 1-7, lines 32-39; Figure 3, element 14; Figure 8, element 42).

45. As to Claim 19, Hollander in view of Chan teach: the one or more waiting requests are for accessing data from the first language portion of the design (Hollander: column 5, lines 44-49; column 9, lines 58-65).

46. As to Claims 20, 36 and 38, Hollander in view of Chan teach: the one or more waiting requests are for debugging the first language portion (Hollander: column 4, lines 45-47; column 5, lines 39-50; column 9, lines 58-65; column 10, lines 43-61).

47. As to Claim 21, Hollander in view of Chan teach: the act of determining whether there are one or more waiting requests for processing of the first language portion is based upon a message sent to a debugger for the processing of the second language portion (Hollander: column 10, lines 24-28 and 50-61; column 11, lines 2-16; Chan: column 7, lines 1-7, lines 32-39; Figure 3, element 14).

48. As to Claim 22, Hollander in view of Chan teach: the act of determining whether there are one or more waiting requests for processing of the first language portion is based a periodic check of a request wait queue for the first language portion (Chan: column 7, lines 1-7, lines 32-46; Figure 3, element 14).

49. As to Claim 23, Hollander in view of Chan teach: the act of determining whether there are one or more waiting requests for processing of the first language portion is

based on whether a threshold number of simulator requests are waiting in a request wait queue (Chan: column 20, lines 55-66; Figure 17, element 89).

50. As to Claim 25, Hollander in view of Chan teach: processing the second language portion comprises debugging the second language portion (Hollander: column 10, lines 24-28, lines 39-42, lines 54-56; column 10, line 67-column 11, line 5; Chan: column 10, lines 37-42).

51. As to Claim 26, Hollander in view of Chan teach: the request processing function is set up ahead of time to handle anticipated requests (Hollander: column 5, lines 44-48; column 11, lines 6-8).

52. **Claims 24, 35 and 37** are rejected under 35 U.S.C. 103(a) as being unpatentable over Hollander in view of Chan as applied to claim 18 above, further in view of Stallman et al.

53. Hollander in view of Chan teach a method for simultaneous processing of a design that is based upon multiple programming languages that includes handling one or more waiting requests for processing of a first language portion by having the processing of a second language portion call a request processing function at the first language portion that has been interrupted.

54. Hollander in view of Chan does not expressly teach: the request processing function is called by a gdb debugger.

55. Stallman et al teaches the gdb, the GNU source level debugger that supports C and C++ among other languages, allows a designer to see what is going on inside a

program while it executes or what a program was doing the moment it crashed and allows a designer to catch bugs in a program by: starting the program and specifying anything that might effect the program's behavior, making the program stop under specified conditions, examining what happened when the program stopped and allowing the designer to experiment with changes to see what effect they have on the program (paragraphs 1-3, bullets 1-4).

56. Hollander in view of Chan and Stallman et al are analogous art since they are directed to the debugging of a general programming language portion of a design.

57. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the handling of one or more waiting requests for processing of the first language portion by having the processing of the second language portion call a request processing function at the first language portion that has been interrupted as taught in Hollander in view of Chan to include the request processing function is called by a gdb debugger since Stallman et al teaches that a gdb debugger allows a designer to see what is going on inside a program while it executes or what a program was doing the moment it crashed and allows a designer to catch bugs in a program by: starting the program and specifying anything that might effect the program's behavior, making the program stop under specified conditions, examining what happened when the program stopped and allowing the designer to experiment with changes to see what effect they have on the program (paragraphs 1-3, bullets 1-4).

Response to Arguments

58. Applicant's arguments filed 10/23/08 have been fully considered but they are not persuasive.

59. Applicant disagrees that Hollander discloses "the HDL" portion and "the general programming language portion". Specifically, although Applicant recites that Hollander does disclose a Verilog model of the design under test (DUT), cited element 172 discloses the "cycle-accurate model of the hardware apparatus", and therefore, 170 and 172 cannot disclose different portions of a same electrical design (page 13, paragraph 2).

The Examiner respectfully disagrees. It is the Examiner's position that Hollander does disclose different portions of a same electrical design as cited above. First, the Examiner would like to point out that Hollander is directed to the verification of an electronic circuit design, and that "design verification is the process of determining whether a particular hardware and software combination…exactly implements the requirements defined by the IC's specification…" (column 1, lines 9-18). These passages set forth that an integrated circuit design has both a hardware "portion" and a software "portion". Further, the cited portions of Hollander, with regards to Figure 5 are directed to co-verification. One of ordinary skill in the art would know that "co-verification" is verifying the hardware design and the software, which is being written to run on the hardware being designed to implement a particular functionality according to the device specifications, at the same time. The Examiner would like to point out that element 163 along with column 10, lines 24-28 was also cited for the limitation of "a

general programming language portion. As described in Hollander, element 170 is an HDL simulation model of the hardware that is being designed, that is the "HDL" or "hardware" portion, and the "external software program", element 168, is the software being written that will run on the device to implement a particular functionality, that is, the "software" portion of the design. Hollander gives an example of co-verification performed to verify a driver program, written in C (column 11, lines 30-36). Although this example sets forth that the software is written in a general programming language, the Examiner also set forth in the 6/23/08 Office Action (paragraph 53), that it is understood that the external program is written in a general programming language (column 10, lines 24-28) wherein the bus-functional model 172, on which the external software program is run is written in Verilog, C, or is a hardware model. Therefore, it is understood that the external software program 163 must be written in a general programming language, such as Verilog or C in order to be run by the bus-functional model 172.

60. Applicant argues that the external software program is "merely another piece of software", Hollander does not disclose that the external software program constitutes a portion of the electronic design, and that the fact that Hollander discloses that the external software program is "compiled and executed by the hardware apparatus while the DUT simulation is run simultaneously" clearly shows that the external software program and the Verilog model are not two portions of the same electronic design (page 13, paragraph 3).

As discussed above, Hollander is directed to the verification of an electronic

circuit design, and that "design verification is the process of determining whether a particular hardware and software combination…exactly implements the requirements defined by the IC's specification…" (column 1, lines 9-18). These passages set forth that an integrated circuit design has both a hardware "portion" and a software "portion". Further, the cited portions of Hollander, with regards to Figure 5 are directed to co-verification. One of ordinary skill in the art would know that "co-verification" is verifying the hardware design and the software, which is being written to run on the hardware to implement a particular functionality according to the device specifications, at the same time. The two "portions" of the design are the hardware and the software. For example, as described in Hollander, element 170 is an HDL simulation model of the hardware that is being designed, that is the "HDL" or "hardware" portion, and the "external software program", element 168, is the software being written that will run on the device to implement a particular functionality. Hollander gives an example of co-verification performed to verify a driver program, written in C (column 11, lines 30-36). It is understood that a "driver" is a software program that interacts with the hardware. It is the Examiner's conclusion that Hollander's disclosure that the external software program is compiled and executed by the hardware apparatus while the DUT simulation is run simultaneously is a teaching of co-verification, and that co-verification is verifying that the hardware and software portions of the same electronic design are functioning appropriately.

61. Applicant argues that Hollander appears to be silent on the apparatus 162 or 166 handling a simulator request which is for the simulation of the HDL portion, And that

Hollander merely discloses the generation of tests for the inputs of both the DUT and co-verification module but never discloses that the co-verification module or any part therein handles the simulator request for the simulator" (page 14, last paragraph).

The Examiner respectfully disagrees. Hollander discloses that the invention, 166, "...can drive 32 or sample 24 any simulator 36 signals...Verilog tasks or VHDL procedures can also be called..." (column 9, lines 58-65). It is the Examiner's interpretation that "driving" a simulator signal and calling Verilog tasks or VHDL procedures teaches "handling" simulator requests for the simulator since these actions will invoke action at the simulator and elicit a response from the simulator (column 10, lines 49, "…the results are returned to the external program". Further, the invention, 166, includes a debugger (column 11, lines 2-5, "...the invention's source-level debugger"). Further, Hollander teaches handling a simulator request (column 10, lines 10 lines 43-50) wherein a co-verification request is sent from the external program to the invention, the invention interprets this request then "executes the appropriate functions" and returns the results to the external program. It is understood that this "executing" the appropriate functions is the "driving", "sampling" or calling procedures at the simulator a discussed (column 9, lines 58-65). Therefore, it is understood that the invention receives the requests from the external program to interact with the simulator, and then drives the signals or calls functions at the simulator, thereby "handling" simulator requests for the simulator. Further, the invention, 166, that handles these simulation requests includes a debugger (column 11, lines 2-5).

62. Applicant disagrees that the cited sections of Hollander (column 5, lines 44-48, column 9, lines 12-13, column 10, lines 35-36 and lines 50-56) disclose "interrupting a simulator that operates upon the HDL portion...by an external debugger to debug the general programming language portion of the electronic design of claim 1 (pages 15-17).

The Examiner notes that Figure 6, element 90, “stop on errors”, “breakpoints”; column 10, lines 24-28, lines 44-50, lines 50-61 and column 11, lines 2-16 were also cited for these limitations.

As to the citation of column 5, lines 44-48, Applicant argues that setting breakpoints in an object-oriented programming language code has no bearing on "interrupting a simulator that operates upon the HDL portion of the electronic design...by an external debugger". The Examiner respectfully disagrees. The cited section is discussing interactive debugging, wherein the user can set breakpoints and observe and change variable values inside the HDL code. It is known in the art that a breakpoint is a point in a computer program wherein execution can be suspended. Therefore, if the HDL design is being debugged, a breakpoint in the code run by the debugger will cause an interruption of the execution of the HDL simulation in order to observe the variables inside the code. Further, as will be discussed below, column 10, lines 44-50 is cited in which, when the external software wants to interact with the DUT, a request is sent to the invention. The invention then drives simulator signals or calls Verilog tasks or VHDL procedures as discussed in column 9, lines 59-65 and returns results to the external program. It is understood that the current execution of the DUT will be interrupted so

that it can service a simulation request from the external software, sent through the invention.

As to the citation of column 9, lines 11-12, Applicant argues that these passages remain silent on and has no bearing on "interrupting the simulator…by an external debugger". These citations were set forth to show further teachings of debugging the DUT. As stated above in regards to column 5, lines 44-48, interactive debugging is discussed, wherein the user can set breakpoints and observe and change variable values inside the HDL code. It is known in the art that a breakpoint is a point in a computer program wherein execution can be suspended. Therefore, if the HDL design is being debugged, a breakpoint in the code run by the debugger will cause an interruption of the execution of the HDL simulation in order to observe the variables inside the code. Further, as will be discussed below, column 10, lines 44-50 is cited in which, when the external software wants to interact with the DUT, a request is sent to the invention. The invention then drives simulator signals or calls Verilog tasks or VHDL procedures as discussed ion column 9, lines 59-65 and returns results to the external program. It is understood that the current execution of the DUT will be interrupted so that it can service a simulation request from the external software, sent through the invention.

As to the citation of column 10, lines 35-36, Applicant argues that these passages remain silent and has no bearing on the claimed invention of "interrupting the simulator…by an external debugger". The Examiner cited this passage to further show that the invention interfaces with the DUT. The Examiner has also revised the citation to

include column 10, lines 44-50 (previously cited for the limitation "the simulator interrupted by an external debugger"). As discussed above, column 10, lines 44-50 describes that when the external software wants to interact with the DUT, a request is sent to the invention. The invention then drives simulator signals or calls Verilog tasks or VHDL procedures as discussed in column 9, lines 59-65 and returns results to the external program. It is understood that the current execution of the DUT will be interrupted so that it can service a simulation request from the external software, sent through the invention.

As to the citation of column 10, lines 50-56, Applicant argues that these passages remain silent and has no bearing on the claimed invention of "interrupting the simulator…by an external debugger". The Examiner cited this passage to further point out that debugging is performed on both the hardware and software sides. As discussed above with regards to column 5, lines 44-48, interactive debugging is discussed, wherein the user can set breakpoints and observe and change variable values inside the HDL code. It is known in the art that a breakpoint is a point in a computer program wherein execution can be suspended. Therefore, if the HDL design is being debugged, a breakpoint in the code run by the debugger will cause an interruption of the execution of the HDL simulation in order to observe the variables inside the code. Further, as will be discussed below, column 10, lines 44-50 is cited in which, when the external software wants to interact with the DUT, a request is sent to the invention. The invention then drives simulator signals or calls Verilog tasks or VHDL procedures as discussed in column 9, lines 59-65 and returns results to the external program. It is understood that

the current execution of the DUT will be interrupted so that it can service a simulation request from the external software, sent through the invention.

As to the citations of Figure 6, element 90 and column 10, lines 44-50, Applicant argues that these limitations are limited between the co-verification extension module and the test generation apparatus and has nothing to do with "interrupting the simulator that operates on the HDL portion…by an external debugger to debug the general programming language portion..." and that the showing of "stop on errors" in the Debug menu of Figure 6 does not disclose the limitation of "interrupting the simulator that operates on the HDL portion…by an external debugger to debug the general programming language portion...". The Examiner respectfully disagrees. As discussed above, column 10, lines 44-50 discussed that when the external software wants to interact with the DUT, a request is sent through the socket to the invention. The invention then drives simulator signals or calls Verilog tasks or VHDL procedures as discussed in column 9, lines 59-65 and returns results to the external program. It is understood that the current execution of the DUT will be interrupted so that it can service a simulation request from the external software, sent through the invention. It is the Examiner's interpretation that the communications are not limited between the co-verification extension module and the test generation apparatus as argued by Applicant, in fact, the test generation apparatus and the extension module allow interactions between the external software and the DUT. Further, Figure 6, element 90 was cited to show an interactive debugging GUI that includes a menu where breakpoints can be set and the options to "continue" or re-start the debugger after execution has been halted.

As to the external debugger debugging the general programming language portion, the Examiner further cited column 11, lines 2-16, wherein it is set forth that the external software (such as, for example, a driver program) is debugged "using the invention".

63. Applicant argues that claims 8-11 are patentable since Hollander does not disclose all the claimed limitations of claim 1 and since Hollander and Chan, either alone or combined, do not disclose, teach or suggest all the limitations of claims 8-11.

The Examiner respectfully disagrees. It is the Examiner's position that Hollander discloses the claimed limitations of claim 1 as discussed above.

64. Applicant argues that claim 14 is patentable since Hollander does not disclose all the claimed limitations of claim 1 and therefore, Hollander and Stallman either alone or combined, do not disclose, teach or suggest all the limitations of claim 14.

The Examiner respectfully disagrees. It is the Examiner's position that Hollander discloses the claimed limitations of claim 1 as discussed above.

65. As to claims 18-23, 25-26, 29-30, 36 and 38, Applicants argues that Hollander and Chan, either alone or combined, do not disclose “the first language portion of the design” and the “second language portion of the design” for similar reasons presented in subsection II-A above (page 19, “A"). Further, Applicant argues that Hollander and Chan, either alone or combined, do not disclose “handling the one or more waiting requests for processing of the first language portion by having processing of the second language portion calla request processing function at the first language portion that has been interrupted, at least one or the one or more waiting requests for processing of the

first language portion causes the processing of the first language portion" for similar reasons presented in subsections II-B and II-C (pages 19-20, "B").

The Examiner has responded to these arguments above. It is the Examiner's position that Hollander in view of Chan teaches or suggests these limitations.

Conclusion

66. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

67. Gay (US Patent 6,188,975) teaches the use of breakpoints in a debugging program to signify to a hardware software co-verification tool when to redirect hardware calls to a hardware simulator.

68. Bian (US Patent 7,117,139) exchanging parameters between a Verilog/PLI module that is co-simulated with a SystemC module using a remote procedure call.

69. Profit, Jr. (US Patent 5,911,059) teaches a communications interface controlling communications between a processor emulator and a hardware simulator.

70. Yunt et al (US Patent 7,464,373) teaches a method and system of exposing debugging information in a graphical modeling and execution environment, and teaches that the debugger will interrupt the simulation when a breakpoint is detected.

71. Bian et al ("VIDE: A Visual VHDL Integrated Design Environment", Proceedings of the ASP-DAC'97, 28-31 January, 1997, pages 383-386) teaches a graphical object-oriented design and debugging approach, wherein, in the debugger, graphical objects

can be specified as interruption points to control the simulation process, and that the simulation is interrupted by a break point.

72. Martionelle et al ("Mixed Language Design Data Access: Procedural Interface Design Considerations", VHDL International Users Forum Fall Workshop, 2000, Proceedings, 10/18-10/20/2000, pages 95-99) teaches different approaches to develop a Verilog/VHDL language interface, and teaches that there are interruption points that interrupt the simulation.

73. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary C. Jacob whose telephone number is 571-272-6249. The examiner can normally be reached Tuesday-Thursday, 7AM-4PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Paul Rodriguez can be reached on 571-272-3753. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Mary C Jacob/
Examiner, Art Unit 2123

/M. C. J./
12/17/08